A Case for Evaluating Sensor Network Protocols Concurrently

Omprakash Gnawali
Stanford University
gnawali@cs.stanford.edu

Leonidas Guibas
Stanford University
guibas@cs.stanford.edu

Philip Levis
Stanford University
pal@cs.stanford.edu

ABSTRACT

Researchers typically evaluate and compare protocols on the testbeds by running them one at a time. This methodology ignores the variation in link qualities and wireless environment across these experiments. These variations can introduce significant noise in the results. Evaluating two protocols concurrently, however, suffers from inter-protocol interactions. These interactions can perturb performance even under very light load, especially timing and timing sensitive protocols. We argue that the benefits of running protocols concurrently greatly outweigh the disadvantages. Protocols rarely run in isolation in real networks, and so considering such interactions is valuable. Although the wireless environment is still uncontrolled, concurrent evaluations make comparisons fair and more statistically sound. Through experiments on two testbeds, we make the case for evaluating and comparing low data-rate sensor network protocols by running them concurrently.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General

General Terms

Experimentation, Performance, Measurement

Keywords

Testbed, Collection, Dissemination, CTP, Sensor Network

1. INTRODUCTION

Wireless testbeds provide an opportunity to test network protocols under realistic wireless environment. Unlike simulations, testbed evaluations do not make simplifying assumptions about the wireless channel. Hence, we can expect the performance of a protocol and its response to dynamics and failures in the real-world deployment to reflect, to some extent, the observations in the testbed experiments.

WiNTECH'10, September 20, 2010, Chicago, Illinois, USA.

Due to the advantages over simulations, wireless testbed evaluations are increasingly perceived as necessary for a serious evaluation of a practical protocol. Many wireless testbeds are setup to support specific wireless research projects [7] and some are even public access [24].

Unlike simulation, however, testbed results are rarely repeatable or reproducible. As real-world phenomena such as human activity, humidity, and external interference all affect the wireless channel, two identical experiments at different times can observe significantly different results. Even back-to-back experiments are not guaranteed to have run on the same set of conditions.

Typically, experiments compare protocols by running each one individually on a testbed. The hope is that performing a large number of experiments and averaging the results will factor out the impact of dynamics on protocol performance. Still, there are several shortcomings to this approach. It requires a large number of experiments while we already face limited availability of time on the testbeds. Most importantly, we are not able to perform a systematic comparison of protocols under the same set of dynamics.

The basic challenge is that wireless protocols are often evaluated in how well they use channel capacity (e.g., throughput). Sensor networks and low-power wireless networks, however, usually operate well below channel capacity. The lower workloads in such networks mean that protocols can often run concurrently without significantly perturbing the wireless channel. Despite this fundamental difference, low-power wireless protocol research still typically follows the well-established methodologies used for high performance networks.

We propose a different approach: experiments should evaluate and compare multiple protocols by running them concurrently on the same testbed. This approach has two advantages. First, experiments will complete in shorter time and allow us to perform more experiments. Unlike simulators that can be instantiated multiple times concurrently, time-sharing is the only way to increase the number of experiments on the testbeds. Second, the protocols are subjected to the same dynamics during the experiment. This property ensures that the protocols and parameters are compared fairly.

Despite these advantages, this methodology is rarely used in wireless research. This lack of adoption is partly due to our lack of understanding of the consequences of running multiple protocols at the same time. For example, small cross-interaction between the protocols could dramatically change the protocol performance on wireless networks. To

the best of our knowledge, there have been no comparative study of these two methodologies. How different are the results between isolated and concurrent runs? What are the pitfalls?

We study the two methodologies, serial experiments and concurrent experiments, in wireless protocol evaluation. We focus on protocols in low data-rate sensor networks. We make these contributions:

- We show that, for certain protocols, running them concurrently on testbeds yields similar results as if they were run serially.
- We present the cases where the results from serial and concurrent experiments are different. We discuss possible reasons for this difference.
- We show that simultaneous execution of protocols provides a more systematic comparison of the protocols than with sequential executions, especially during link failures and adverse dynamics.
- We show that the concurrent methodology is also applicable to comparing the impact of different parameter values of a protocol on its performance.

2. EXPERIMENT METHODOLOGIES

In this section, we describe the two methodologies used in network protocol (L3) experiments on wireless network testbeds.

2.1 Serial Experiments

Serial experiments run protocols one at a time on the testbeds. The researchers perform a large number of experiments to factor out the effects of network dynamics on protocol performance across the experiments. This is the most common experimental methodology used by the researchers.

In a recent study that compared a network protocol called CTP [11] against a proposed Bursty Routing Extension [5] to CTP, the authors performed a series of 30-minute experiments, running one protocol at a time on the Motelab testbed [24]. Another study comparing broadcast protocols ran two protocols one after another on a wireless testbed [21]. We present these examples, not to single out the methodologies used in these papers, but to illustrate the norm in wireless testbed experiments.

The main advantage of this approach is that the protocol has precise control over the bandwidth used and packet timing due to exclusive use of the radio. We can guarantee that cross-interaction with other protocols did not change the protocol performance. Two main drawbacks of this approach are requiring exclusive access to the testbed for long period of time, and not being able to ensure that the protocol comparisons are performed on the same set of network dynamics and channel quality. When the wireless environment changes, so does the experiment environment.

2.2 Concurrent Experiments

Concurrent experiments run multiple protocols simultaneously on each node of the wireless testbeds. All the protocols in a concurrent experiment are subjected to the same wireless environment, with or without dynamics. If the link qualities degrade, all the protocols must work with the same degraded links for the same duration of time. This approach is one of the best ways to compare the performance of protocols under network dynamics.

Although it is infeasible to run hundreds of protocols at the same time, concurrent experiments are useful in wireless research. Typically, we compare a proposed protocol or solution with 1-5 other protocols or settings. This level of concurrency is achievable even on resource-constrained wireless sensor networks. We run up to three protocols concurrently in our experiments and it is common for a sensor network deployment to run a handful of protocols (collection, dissemination, time synchronization, etc.).

Although the required level of concurrency is achievable, concurrent experiments are not widely used in wireless research. This lack of adoption is partly due to the lack of systematic study of the effectiveness and pitfalls of this approach. This is also due to the inapplicability of this approach for certain types of experiments:

- Experiments that involve high data rates, with aggregate rate near or above channel capacity. If the protocols compete with each other for bandwidth, the experiments capture not only the performance of the protocols on wireless network but also its interaction with other protocols.
- Synchronized bursts. If the protocol combinations generate synchronized bursts of control or data packets, the performance will likely degrade despite the aggregate load is well below the channel capacity.
- Timing-sensitive experiments. The performance of some protocols depend on timing of messages at a fine scale. For example, protocol timeouts that are triggered by a few extra milliseconds of delay due to pending packet from a concurrently running protocol can alter protocol performance.
- Experiments that change certain link and physical-layer attributes. The notion of wireless neighborhood and link qualities can be fundamentally altered when one of the protocols in a concurrent experiment changes link or physical layer parameters. Examples include changes to the sleep interval of a link layer protocol [18], topology control by changing the transmit power [13] or receiver gain, and changing the wireless channel by a frequency-hopping protocol [23]. Such changes can negatively impact the performance of concurrently running protocols.

We focus on the evaluation of low data rate network protocols on top of CSMA link layer. In concurrent experiments with such network layer protocols, we assume a coarse grained multiplexing in the channel: the space between the packets from the different protocols is typically much larger than the channel coherence time.

3. EVALUATION

In this section, we describe our study of the serial and concurrent approaches to wireless network experiments. We first explain the experimental setup and then describe the results.

3.1 Testbeds, Protocols, and Metrics

We perform our experiments on two wireless testbeds. Tutornet [4] has 54 TelosB [19] nodes on the 3rd floor of the RTH building at USC. TelosB nodes have a 4 MHz MSP430 processor, 10 KB of RAM, come with CC2420 802.15.4-compatible radio, and run TinyOS 2.x [16]. Thus, they are ideal for low-power wireless experiments. Motelab [24] has 100 operational TelosB nodes across the three floors of an Engineering building at Harvard. Motelab testbed has many links that are intermittent and often presents a challenging test for network protocols. We collect the statistics from the experiments using the wired backchannel available on these testbeds. We ran the experiments on 802.15.4 channel 26, except for one experiment in Section 3.6. Channel 26 does not overlap with the frequencies used by WiFi access points in the building. We ran each experiment five times and present the averages.

For our experiments, we pick two classes of best-effort network protocols common in low power wireless sensor networks – collection and dissemination.

Collection protocols compute the shortest paths from all the nodes in the network to a single or a small number of destinations, also called roots, in the network. For example, a network of wireless sensors can use a collection protocol to report their readings to a central server. CTP [11] selects paths with the minimum ETX [9] metric. MultihopLQI [2] uses the LQI metric, which is a link quality indicator provided by the CC2420 radio [15], to find the best routes to the root. The main metrics used to evaluate the collection protocols are:

- *Delivery ratio* is the ratio of packets received at the root to the packets sent by the nodes in the network. High delivery ratio, close to 100%, is desirable.
- *Delivery cost* is the number of packet transmissions required per successfully delivered packet. Low delivery cost is desirable.
- *Path length* is the average number of hops travelled by the packets to the root.
- *Churn* is the number of times a node changes its next hop.

Dissemination protocols, similar to multicast protocols, are used to send the same information to all the nodes. For example, a central server can use a dissemination protocol to send a command to a network of wireless sensors to start sampling. Drip [1] is a basic dissemination protocol. Compared to naive flooding, Drip achieves efficiency by avoiding redundant transmissions if the same information has already been received by the nodes in the neighborhood through different relay nodes. Compared to Drip, DIP [17] and DHV [8] can scale to a large number of data item updates, however perform worse than Drip on small number of data items or updates. The main metrics used to evaluate dissemination protocols are:

- *Dissemination reliability* is the ratio of updates received to the updates sent. Dissemination reliability close to 100% is desirable.
- *Dissemination cost* is the number of transmissions for each packet dissemination per node. This cost can be less than 1 because one broadcast transmission can deliver packets to multiple neighbors.

3.2 Serial Experiments

We ran the two collection protocols and the three dissemination protocols one at a time. These experiments establish a baseline performance for each protocol. Later, we compare these results to the results obtained from concurrent experiments.

In each collection experiment on Motelab and Tutornet, all the nodes send one packet every 8s to the root. Packet send times include jitter so that packets are not synchronized across the network. In each dissemination experiment, one node sends a new update to a key to the network every 30s. Each collection and dissemination experiment runs for one hour.

CTP and MultihopLQI achieved 99.9%+ packet delivery on Tutornet. On Motelab, CTP achieved 96.3% packet delivery while MultihopLQI achieved 93.6% packet delivery. Drip, DIP, and DHV achieved 99.9%, 99.3%, and 96.7% dissemination delivery on Tutornet. They achieved 99.8%, 98.6%, and 86.5% dissemination delivery on Motelab. In the next section, we compare these baseline performances with the results from the concurrent experiments.

3.3 Concurrent Collection Experiments

For concurrent collection experiments, we ran CTP and MultihopLQI on the nodes at the same time. Each node generated two data packets every 8s, sent one using CTP, and the other using MultihopLQI. A total of two packets every 8s from each node is well below the channel capacity on both the testbeds. We added jitter to the packet timing to prevent synchronized bursts.

Table 1 presents the main results for the collection protocols. We found that delivery ratio achieved by MultihopLQI and CTP when they are run concurrently remain within 1.08% of what each achieved individually, both on Tutornet and Motelab. The Cost, Path Length, and Cost/Path Length (PL) metrics are different by up to 11% between serial and concurrent experiments. The Churn/node-hr metric shows the biggest variation (up to 180%) between serial and concurrent experiment.

These experiments show that different performance metrics change to different extent between serial and concurrent experiments. We make two observations that offer a possible explanation for this difference. First, protocols are designed to optimize some aspect of the performance. Hence, even with channel perturbation or cross-interaction in a concurrent experiment, the protocols still achieve what it achieved in a serial experiment. For example, CTP is designed to achieve high delivery ratio. It achieves high delivery ratio regardless of the other protocols running concurrently. Hence, CTP's performance measured through the delivery ratio metric changes little between serial and concurrent experiments.

The second observation is, the large variation in performance across serial and concurrent experiments can be predicted by the large variation within serial experiments. Figure 1 shows that the variation in Delivery, Cost, and Path Length metrics are small within serial experiments. Churn shows larger variation within serial experiments. Hence, the large discrepancy in Churn between serial and concurrent experiments is due to the dynamic aspect of the protocol

Protocol	Experiment	Testbed	Delivery	Cost	Path Length	$\frac{\text{Cost}}{\text{Path Length}}$	$\frac{\text{Churn}}{\text{node-hr}}$
LQI	LQI alone	Tutornet	99.91%	2.12	1.82	1.16	5.42
LQI	CTP+LQI	Tutornet	99.98%	2.23	1.80	1.24	5.48
CTP	CTP alone	Tutornet	100.00%	2.22	2.14	1.04	0.28
CTP	CTP+LQI	Tutornet	99.99%	2.03	1.93	1.05	0.10
LQI	LQI alone	Motelab	93.60%	4.80	3.21	1.49	9.76
LQI	CTP+LQI	Motelab	92.60%	5.30	3.40	1.56	10.52
CTP	CTP alone	Motelab	96.35%	7.03	3.47	2.03	5.85
CTP	CTP+LQI	Motelab	96.38%	7.29	3.81	1.92	4.64

Table 1: Performance of CTP and MultihopLQI (LQI) in serial and concurrent experiments.

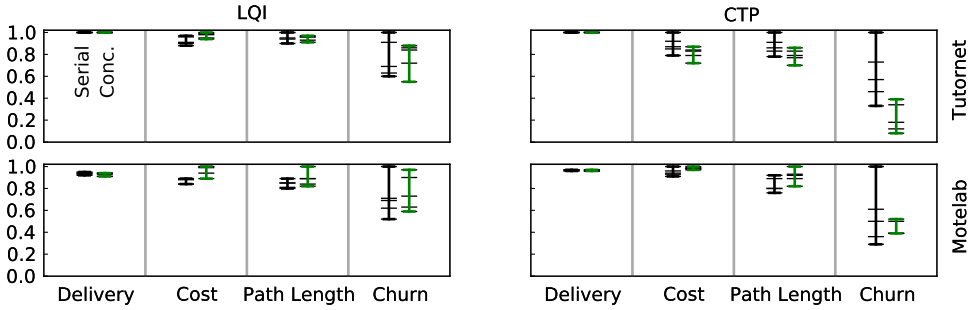

Figure 1: Result of Serial and Concurrent collection experiments on Tutornet and Motelab. Cost, Path Length, and Churn are normalized to the maximum value observed for that metric. Each bar shows the observations from five runs. Compared to other metrics, Churn shows the highest variation across the experiments.

(opportunistic parent selection by CTP, for example) measured by this metric rather than due to serial or parallel experimentation.

3.4 Concurrent Dissemination Experiments

For concurrent dissemination experiments, we ran two sets of experiments, first with Drip+DIP (Conc-2) and second with Drip+DIP+DHV (Conc-3) combinations, both on Tutornet and Motelab. Each node generated a new update every 30s and sent it to the network separately using each dissemination protocol concurrently. The load generated by these experiments is well below the channel capacity on the testbeds.

Table 2 summarizes the results from our dissemination experiments. We found that the performance of Drip, when we run it by itself, or with DIP, or with both DIP and DHV, is similar. The difference in dissemination reliability across the combinations and testbeds is within 0.23% and Update Cost within 1.04%. The dissemination reliability achieved by DIP and DHV is different by up to 13.45% between serial and concurrent experiments across the testbeds. The Cost per Update metric for DIP and DHV, however, changes significantly over the experiments. For example, DHV's Cost per Update increases by 39.97% when it is run concurrently with Drip and DIP compared to its Cost per Update when it is run alone on Tutornet.

Similar to the observation we made about collection protocols, the metrics that showed the most variance within serial experiments also showed the biggest change between serial and concurrent experiments. Figure 2 shows that Drip's dissemination reliability does not change across serial experiments on Tutornet. As expected, we did not observe any change in its reliability between serial and concurrent experiments. On Motelab, DHV's reliability showed the largest variation compared to Drip and DIP. The difference in DHV's reliability between serial and concurrent experiment was also high (39.97%). These results suggest that they are likely experimental variations (because the protocol does not optimize these metrics) rather than consequences of concurrent experiments.

3.5 Pathological Experiments

In Section 2.2, we discussed a few scenarios in which protocol experiments can cross-interact and are not appropriate for concurrent experiments. We now explore two such scenarios.

On Tutornet, we ran CTP and MultihopLQI concurrently. The application using CTP tried to send 2 packets every second. The application using MultihopLQI tried to send 1 packet every 8 seconds. Due to the high load from the CTP packets, MultihopLQI rarely finds a free channel to transmit its packets. It drops many packets as the queues fill up. MultihopLQI achieved a delivery ratio of 72% at a cost of 8.26 packet transmissions for each packet delivery, about 4x the cost in serial experiments. This poor performance is not due to the design flaws, but rather a result of interaction with another protocol experiment. Thus, when planning and performing concurrent experiments, we need to consider the parameters and configurations of all the protocols.

In another experiment, we ran CTP and MultihopLQI concurrently on Motelab. We removed the jitter in the packet send time so that the two packets, one with CTP

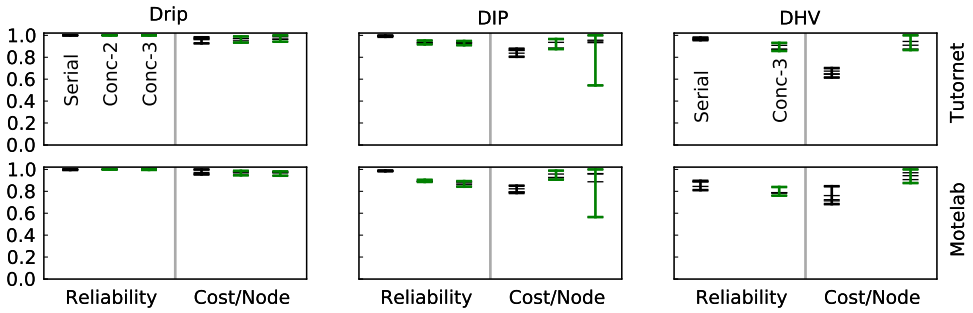

Figure 2: Result of Serial and Concurrent dissemination experiments, Conc-2 (Drip+DIP) and Conc-3 (Drip+DIP+DHV), on Tutornet and Motelab. Cost/node is normalized to the maximum value observed. Each bar shows the observations from five runs. Cost/node shows higher variation than Reliability in serial and concurrent experiments.

Protocol	Experiment	Reliability	Cost / Node
	Tutornet Experiments		
Drip	Drip alone	99.99%	0.21
Drip	Drip+DIP	100.00%	0.21
Drip	Drip+DIP+DHV	100.00%	0.22
DIP	DIP alone	99.31%	0.36
DIP	Drip+DIP	93.42%	0.39
DIP	Drip+DIP+DHV	93.00%	0.37
DHV	DHV alone	96.68%	0.34
DHV	Drip+DIP+DHV	88.90%	0.48
	Motelab Experiments		
Drip	Drip alone	99.82%	0.86
Drip	Drip+DIP	99.99%	0.85
Drip	Drip+DIP+DHV	100.00%	0.85
DIP	DIP alone	98.56%	1.41
DIP	Drip+DIP	89.27%	1.63
DIP	Drip+DIP+DHV	87.18%	1.52
DHV	DHV alone	86.52%	1.89
DHV	Drip+DIP+DHV	79.18%	2.38

Table 2: Performance of Drip, DIP, and DHV in serial and concurrent experiments.

	CTP	MultihopLQI
Delivery	95.3%	85.0%
Cost	8.43	6.45
Path Length	4.05	3.39
$\frac{\text{Cost}}{\text{Path Length}}$	2.08	1.90
$\frac{\text{Churn}}{\text{node-hr}}$	6.50	2.63

Table 3: Performance of CTP and MultihopLQI on Motelab with synchronized packets.

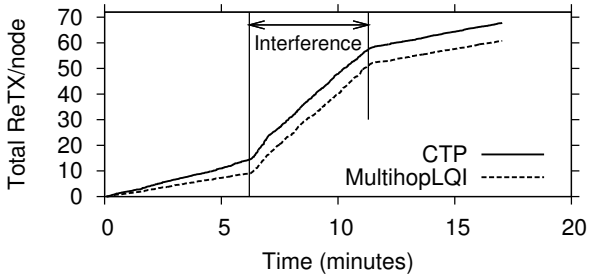

Figure 3: Concurrent CTP and MultihopLQI under controlled interference. Both protocols incur larger number of retransmissions during the 6-minute period with interference.

and one with MultihopLQI, were sent as close to each other as possible, and from all the nodes in the testbed. Although the aggregate load is below the channel capacity, this timing pattern creates a burst of packets in the network and cause packet drops. Table 3 summarizes the results. Both the protocols achieved lower delivery ratio and higher delivery cost than in serial experiments. When protocols tightly control the timing of their packets, there is a possibility of such packet synchronization across the protocols in a concurrent experiment. Such packet synchronization results in bursts of packet and degraded performance, even though those protocols would have achieved better performance in serial experiments.

3.6 Network Dynamics

To use concurrent experiments to evaluate protocols under the same dynamics, we perform two experiments.

First, we perform an experiment with controlled dynamics. We ran CTP and MultihopLQI concurrently on Tutornet, with each node sending 1 packet every 8s packets using each protocol. During the 6-12 minute interval, we introduced unwanted interference by having four nodes transmit back-to-back broadcast packets. CTP and MultihopLQI still achieved near 100% delivery ratio during this period however at a higher cost. Figure 3 shows the number of packet retransmissions for CTP and MultihopLQI. We can see much larger number of retransmissions during this period. When the interfering nodes are turned off, the number of retransmissions drop for both the protocols to the normal level. During this experiment both the protocols were subjected to the same dynamics and we were able to observe the performance and reaction of the two protocols.

In our next experiment, we ran CTP and MultihopLQI concurrently on Tutornet for 12 hours on channel 16. Chan-

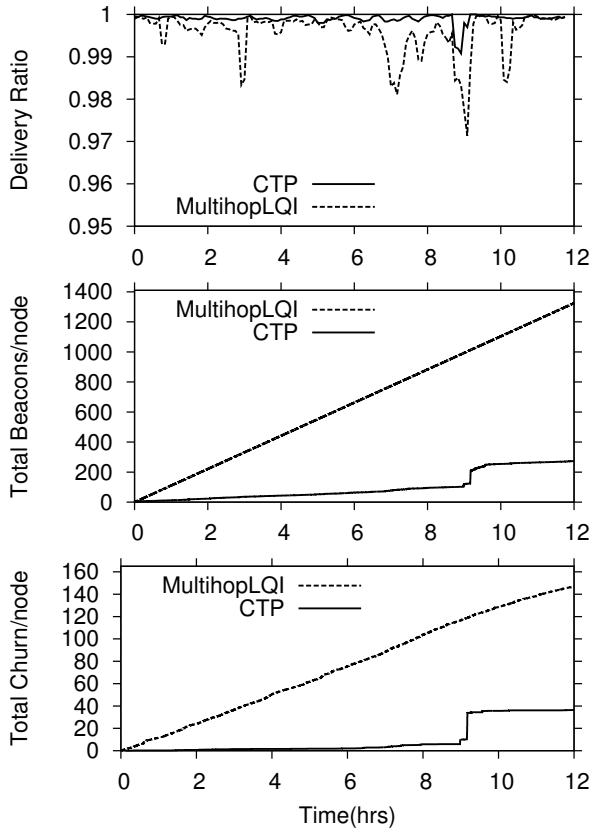

Figure 4: Concurrent CTP and MultihopLQI under dynamics. During disruption (just before 9th hr), CTP's route repair mechanism reacts aggressively and recovers quickly, hence jump in beacons and churn. MultihopLQI repairs the topology at the normal discovery rate.

nel 16 overlaps with WiFi channel used in the building and can experience disruptions when WiFi traffic volume is high. Figure 4 shows the delivery ratio achieved by these two protocols, averaged every 15 minutes. Out of four regions with delivery ratio less than 98% for MultihopLQI, CTP was affected during the third region, although to a lesser extent. Furthermore, CTP recovered in half the time it took MultihopLQI to recover. The same figure also shows the churn and control packets. CTP's churn and control packets increase during this disruption as it tried sending packets on alternative nexthop nodes to recover from the disruption as soon as possible. MultihopLQI's control overhead and churn does not change much because it uses periodic beaconing and route discovery. This result contrasts the recovery mechanisms used in CTP and MultihopLQI when they are subjected to the identical network dynamics. Thus, with concurrent experiments, we have the ability to study how two protocols respond to the same dynamics.

3.7 Protocol Parameters

Concurrent experiments can also be used to study the impact of different parameter values on protocol performance. As a case study, we changed the path switching threshold in CTP from ETX=1.5 to ETX=2.5. With the new threshold, CTP chooses a new path only if its ETX is smaller than the ETX of the current path by at least 2.5. This change is expected to make CTP more stable. We concurrently ran these two versions of CTP on Tutornet, each version of CTP delivering 1 packet every 8s from each node in the network.

Route switch threshold	**1.5 ETX** (default)	**2.5 ETX**
Delivery	99.9%	99.9%
Cost	2.13	2.69
Path Length	1.81	2.30
$\frac{\text{Cost}}{\text{Path Length}}$	1.18	1.17

Table 4: Key results to study the impact of larger route switch threshold on CTP's performance.

Table 4 summarizes the key results from this experiment. Increasing the threshold from 1.5 ETX to 2.5 ETX increased the packet delivery cost from 2.13 to 2.69 transmissions, an increase of 26%. The remaining two metrics explain why. The paths are longer with the larger threshold. CTP will switch to the new path only if it is significantly (2.5 ETX) better and thereby not using marginally better (and shorter) paths to achieve lower delivery cost. Interestingly, the Cost/PL metric is similar with both the thresholds suggesting CTP continues to use high quality links that require few retransmissions.

This approach of studying the protocol parameters gives us the confidence that the parameter values for a protocol are being evaluated on the same set of dynamics and hence indicative of the true impact of the parameter and not the dynamics in the network.

4. RELATED WORK

Wireless networks routinely run many protocols concurrently. For example, a Tenet [12] system runs collection, dissemination, and time synchronization protocols at the same time. CSMA is a classic mechanism used by MAC protocols to arbitrate concurrent communication. Work by Vaidya et al. [22] and protocol isolation [14] propose mechanisms to ensure fairness across the protocols in a wireless network. Such mechanisms will make concurrent execution of protocols more common. Our work is more concerned with understanding the implications of running protocols concurrently in a wireless network.

Within a node, the layered architecture of the network stack allows multiple higher layer components to concurrently use the lower layer services. Multiple network protocols may run on top of the link layer and multiple transport protocols may run on top of a network layer. As a specific example, TinyOS 2.x [16] messaging API guarantees that each packet sender has one slot in the link layer send queue, regardless of the load presented by other packet senders. We advocate parallelizing testbed experiments by running multiple protocols concurrently leveraging the concurrency already supported in such platforms.

In systems and networking research, Planetlab [10] is by far the best known example of a shared testbed where many experiments can run simultaneously. GENI [3] is another networking experiment infrastructure that supports concurrent experiments through slicing. OpenRoads slices can al-

locate network resources to different concurrent wireless experiments [25]. Techniques such as spectrum slicing [6] and wireless virtualization [20] enable other approaches to concurrent experiments. In this paper, we discuss the classes of wireless experiments that can be parallelized on shared testbeds and also explore the differences between serial and parallel experiments in low-utilization sensor network protocols.

5. CONCLUSIONS

Protocol evaluations on wireless testbeds are typically done using a series of experiments, one protocol or configuration at a time. This approach can be significantly more time consuming than simulations. We showed that concurrent execution of network protocols is one possible way to increase the number of experiments on these testbeds. Concurrent execution also enables a fair and systematic comparison of protocols under dynamic environment found on many wireless testbeds. However, there are many pitfalls to this approach. We articulated and studied some of these issues. We hope this will provide hints for experiment design to the researchers as they plan concurrent experiments on wireless testbeds.

Acknowledgments

This work was supported by ARO AHPCRC grant W911NF-07-2-0027, generous gifts from DoCoMo Capital, the National Science Foundation under grants #0626151, #0831163, and #0846014, the King Abdullah University of Science and Technology (KAUST), Microsoft Research, and a Stanford Terman Fellowship.

6. REFERENCES

[1] The Drip protocol. `http://www.tinyos.net/tinyos-2.x/tos/lib/net/drip`, 2009.

[2] The MultiHopLQI protocol. `http://www.tinyos.net/tinyos-2.x/tos/lib/net/lqi`, 2009.

[3] The Global Environment for Network Innovations (GENI). `http://www.geni.net`, 2010.

[4] The USC Tutornet Testbed. `http://testbed.usc.edu`, 2010.

[5] Muhammad Hamad Alizai, Olaf Landsiedel, Jó Ágila Bitsch Link, Stefan Götz, and Klaus Wehrle. Bursty traffic over bursty links. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 71–84, New York, NY, USA, 2009. ACM.

[6] Angelos-Christos Anadiotis, Apostolos Apostolaras, Dimitris Syrivelis, Thanasis Korakis, Leandros Tassiulas, Luis Rodriguez, and Maximilian Ott. A new slicing scheme for efficient use of wireless testbeds. In *WINTECH '09: Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization*, pages 83–84, New York, NY, USA, 2009. ACM.

[7] Benjamin A. Chambers. The grid roofnet: a rooftop ad hoc wireless network. Master's thesis, Massachusetts Institute of Technology, May 2002.

[8] Thanh Dang, Nirupama Bulusu, Wu chi Feng, and Seungweon Park. DHV: A Code Consistent Maintenance Protocol for Wireless Sensor Networks. In *EWSN '09: Proceedings of the 6th 6th European Conference on Wireless Sensor Networks*, Cork, Ireland, 2009.

[9] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146, New York, NY, USA, 2003. ACM.

[10] Marc E. Fiuczynski. Planetlab: overview, history, and future directions. *SIGOPS Oper. Syst. Rev.*, 40(1):6–10, 2006.

[11] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection Tree Protocol. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14, New York, NY, USA, 2009. ACM.

[12] Omprakash Gnawali, Ki-Young Jang, Jeongyeup Paek, Marcos Vieira, Ramesh Govindan, Ben Greenstein, August Joki, Deborah Estrin, and Eddie Kohler. The tenet architecture for tiered sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 153–166, New York, NY, USA, 2006. ACM.

[13] Gregory Hackmann, Octav Chipara, and Chenyang Lu. Robust topology control for indoor wireless sensor networks. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 57–70, New York, NY, USA, 2008. ACM.

[14] Jung Il Choi, Maria A. Kazandjieva, Mayank Jain, and Philip Levis. The case for a network protocol isolation layer. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 267–280, New York, NY, USA, 2009. ACM.

[15] ChipCon Inc. Cc2420 data sheet. `http://www.chipcon.com/files/CC2420_Data_Sheet_1_0.pdf`, 2003.

[16] P. Levis, D. Gay, V. Handziski, J.-H.Hauer, B.Greenstein, M.Turon, J.Hui, K.Klues, C.Sharp, R.Szewczyk, J.Polastre, P.Buonadonna, L.Nachman, G.Tolle, D.Culler, and A.Wolisz. T2: A Second Generation OS For Embedded Sensor Networks. Technical report, Telecommunication Networks Group, Technische Universitat Berlin, 2005.

[17] Kaisen Lin and Philip Levis. Data Discovery and Dissemination with DIP. In *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, pages 433–444, Washington, DC, USA, 2008. IEEE Computer Society.

[18] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM.

[19] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 48, Piscataway, NJ, USA, 2005. IEEE Press.

[20] Gregory Smith, Anmol Chaturvedi, Arunesh Mishra, and Suman Banerjee. Wireless virtualization on commodity 802.11 hardware. In *WinTECH '07: Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pages 75–82, New York, NY, USA, 2007. ACM.

[21] Fred Stann, John Heidemann, Rajesh Shroff, and Muhammad Zaki Murtaza. Rbp: robust broadcast propagation in wireless networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 85–98, New York, NY, USA, 2006. ACM.

[22] Nitin H. Vaidya, Paramvir Bahl, and Seema Gupta. Distributed fair scheduling in a wireless lan. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 167–178, New York, NY, USA, 2000. ACM.

[23] Thomas Watteyne, Ankur Mehta, and Kris Pister. Reliability through frequency diversity: why channel hopping makes sense. In *PE-WASUN '09: Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 116–123, New York, NY, USA, 2009. ACM.

[24] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. Motelab: a wireless sensor network testbed. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 68, Piscataway, NJ, USA, 2005. IEEE Press.

[25] Kok-Kiong Yap, Masayoshi Kobayashi, David Underhill, Srinivasan Seetharaman, Peyman Kazemian, and Nick McKeown. The stanford openroads deployment. In *WINTECH '09: Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization*, pages 59–66, New York, NY, USA, 2009. ACM.